Amendments to the Specification:

Please replace the paragraph that begins on page 2, line 28, with the following amended paragraph:

The isolation, ~~Identification~~ identification and correction of problems encountered during the execution of a computer script are difficult processes. This is true for proven scripts and is particularly a problem in the development and debugging of script files. Execution of a computer script is not an interactive process, and a problem encountered during execution is not readily isolated to a single command within the script. A problem may be encountered during execution due to a logic flaw or command format within the script itself. Another source of script execution problems is the malfunctioning of the computer upon which the script is executing or problems with other hardware associated with that computer. Node ~~Malfunctions or~~ malfunctions and node unavailability are particularly common problems that are encountered when executing a script on multiple computer network nodes. Script execution in a distributed shell environment often causes a large amount of text output from all [[of]] the nodes to be displayed on the operator screen. Error messages are included in this stream and may not be noticed by the operator. The operator may mistakenly assume that all nodes have been properly configured when some have not due to errors that were not noticed by the operator. The unavailability of a node or improper execution on a node of a command within a script being executed in a distributed shell environment on multiple nodes can cause the entire execution to "hang". Isolation of the cause of this malfunction is time consuming since each node must be tested to confirm proper operation. The impact of this is worse during script development where the cause of the problem is assumed to be a logic flaw in the script and not a hardware problem among the executing nodes.

Please replace the paragraph that begins on page 3, line 21, with the following amended paragraph:

Command scripts that contain configuration commands are particularly difficult to debug or rerun in order to identify the source of errors. Configuration commands within a command script may create problems if they are re-executed ~~within~~ while undoing the action of a previous execution of the command. An example of such a configuration command is a command to mount a file system. If a command to mount a file system is executed twice, the same file system will have two mount points. Such multiple executions of configuration commands may induce problems beyond those which caused the initial script failure. Proper isolation of problems encountered during the execution of a script file requires that the effects of commands that were properly executed be undone prior to re-execution of the script in order to attempt to isolate problems. Such undoing of these effects is difficult since the commands that were executed during a failed script execution may not be known. This requires manual identification of the status of the computer node to determine if the status has been altered by commands within the script or a rebooting of the computer as a safety measure.

Please replace the paragraph that begins on page 4, line 13, with the following amended paragraph:

According to one aspect of a preferred embodiment of the present invention, a method for interactive monitoring and control of computer script commands in a network environment includes accepting an operator defined plurality of computer commands that are contained within a command file. The plurality of computer commands are then displayed to the operator in a GUI display that allows the operator to accept ~~accepting~~ a selection of at least one of the commands contained within the plurality of computer commands. The selected commands are then executed.

Please replace the paragraph that begins on page 4, line 21, with the following amended paragraph:

According to another aspect of a preferred embodiment of the present invention, [[an]] a system for interactive monitoring and control of computer script commands in a network environment includes a command list that contains commands to be executed, a command display that is communicatively coupled to the command list and that displays at least one command within the command list. The system further contains a command selector that is communicatively coupled to the command display and that allows selection of a selected command list, wherein the selected command list contains at least one of the commands that are contained within the command list. The system also contains a command dispatcher that is communicatively coupled to the command selector, and that causes the selected command list to execute.

Please replace the paragraph that begins on page 5, line 20, with the following amended paragraph:

The present invention, according to a preferred embodiment, overcomes problems with the prior art by providing a graphical user interface (GUI) to present an operator with a display of commands within a script and means for controlling and monitoring execution of those commands. The GUI further aids the operator to select one or more commands within a command script file for execution, and allows the operator to control execution of a selected subset of commands contained within the script and to monitor the standard commands outputs and error outputs from the commands in separate window panes within the GUI. The command scripts of the preferred embodiment allow the developer of the script to specify an undo command for some or all of the commands within the script, thereby allowing more efficient "trial and error" debugging and problem identification and isolation. The preferred embodiment further allows the operator to specify one or more computer nodes within a network or cluster upon which the commands contained within the script will be simultaneously executed. In a preferred embodiment, an error in execution of the command script in a multiple processor environment will result in execution of commands on each individual processor to aid in the identification of processors ~~which~~ that are not properly responding.

Please replace the paragraph that begins on page 6, line 8, with the following amended paragraph:

A schematic of one exemplary system architecture 100 for a preferred embodiment of the present invention is illustrated in FIG. 1. The exemplary system architecture 100 comprises a GUI interface 106 to allow the operator to monitor and control execution of computer commands. The GUI interface 106 of this exemplary embodiment executes upon a local computer 108. The GUI interface 106 accepts inputs from a command file 102 and a node list file 104. The command file 102 contains a list of computer operating system commands that are to be executed in conjunction with the operation of this exemplary embodiment of the present invention. The operating system commands in the command file 102 are able to include application programs that execute under an operating system and are also able to include operating system commands for direct execution. The node list file 104 contains a list of computer nodes on which to execute the commands contained within the command file 102. The node list file 104 contents can include a single processor node, such as the local computer ~~106~~ 108. The node list file 104 is also able to include one or more other computer nodes that are communicatively coupled to the local computer 108. The local computer 108 of the exemplary embodiment is connected to a computer network 110 ~~which~~ that provides communications between the local computer 108 and one or more remote computers, such as remote computer A 120a and remote computer B 120b, on which ~~computer~~ operating system commands that are monitored and controlled through the GUI interface 106 are able to execute. The local computer 108 also has an operator interface 112 that, in preferred embodiments, comprises a display, a keyboard and a mouse or other pointer control device. The operator interface 112 is able to utilize other operator input and/or output devices as are known to practitioners in the relevant arts.

Please replace the paragraph that begins on page 9, line 8, with the following amended paragraph:

An exemplary GUI display 200 as displayed by an exemplary embodiment of the present invention is illustrated in FIG. 2. The exemplary GUI display 200 contains multiple displays that include windows, or window panes, that display text and allow the operator to scroll through the displayed text to view text above or below that displayed. Data that is referred to herein as displayed "in" a window include, in addition to the data currently displayed in the window, those items which are not currently displayed in the window but which are accessible by scrolling the window by ~~user~~ use of the GUI functions that are available as are known to practitioners in the relevant arts. This exemplary GUI display 200 includes a command file field 202 that indicates the name of the command file 102. In the exemplary embodiment, the command file 102 is a computer data file that contains textual representations of operating system commands in ASCII text format. Alternative embodiments may incorporate command files 102 that are encoded or stored through alternative methods, including storage in ROM or communicated through a variety of communications techniques. The exemplary GUI display 200 also contains a command window 204 that displays a portion of the commands contained within the command file 102. The command window 204 of the exemplary embodiment allows the operator to scroll up and down in order to display different parts of the command file 102. The command window 204 contains a scroll bar 205 ~~which~~ that allows the operator to scroll the contents of the command window by using a pointing device and the slider of the scroll bar 205. The operator is able to also scroll the contents of the command window by using control characters entered from a keyboard that is part of the operator interface 112. Control characters are associated with special function keys in the exemplary embodiment, such as the "page up" and "page down" keys on the keyboard that is part of the operator interface 112. ~~The command window 102, along with the operator interface 112,~~ The command window 204 along

with the operator interface 112 of the exemplary embodiment provide a command selector that allows the operator to select one or more commands listed within the command window ~~102~~ 204. The operation of the exemplary embodiment allows the selected commands, or selected command list, to be executed either singularly or as a group. Commands are selected in the exemplary embodiment through the use of GUI operations that include placing a pointer, ~~that~~ which is controlled by movement of a mouse input device of the GUI interface, over the command ~~to select and~~ to be selected, and clicking a mouse button to select the command ~~which~~ that is under the GUI display pointer.

Please replace the paragraph that begins on page 11, line 5, with the following amended paragraph:

The command window 204 of the exemplary embodiment allows the operator to ~~select~~ see one or more of the commands that are displayed within the command window 204. [[A]] The command selector of the exemplary embodiment allows commands to be selected through the various techniques implemented in GUI interfaces to select multiple items in a display window, as are known to practitioners in the relevant arts. The Select All function button 218 in the exemplary GUI display 200 allows the operator to select all [[of]] the commands that are listed within the command window, including those commands not currently displayed but are accessible by scrolling through the command window 204. Pressing the Select All function button 218 has the effect of selecting each command contained within the command file 102.

Please replace the paragraph that begins on page 11, line 17, with the following amended paragraph:

The Deselect All function button 222 has the effect of deselecting all [[of]] the commands ~~which~~ that are selected within the command display window 204. The commands ~~which~~ that are selected may have been selected by any means supported by the particular embodiment.

Please replace the paragraph that begins on page 11, line 22, with the following amended paragraph:

Selecting the Do All function button 232 triggers [[the]] a command dispatcher of the exemplary embodiment ~~to causes all of the commands which~~ to cause all the commands that have been selected, i.e., the selected command list, to execute on the computer nodes displayed within the node window 214. ~~The set of commands that are selected are able to be a subset of the commands listed in the command file 102 as well as all of the commands.~~ The set of commands that are selected can be either a subset of the commands, or all the commands, listed in the command file 102. The commands that are selected ~~are also able to~~ can be either contiguous or non-contiguous commands within the command file 102. Non-continuous commands are selected by using the GUI facilities to select non-contiguous commands, and then these commands are executed by pressing the Do All function button 232. Methods to select non-contiguous entries in a GUI window are known to practitioners and include pressing a keyboard key, such as the 'control' key, during selection of multiple, non-contiguous entries. Execution of non-contiguous commands in the exemplary embodiment of the present invention affords several advantages in the development, debugging and maintenance of scripts. Execution of non-contiguous commands allows commands to be skipped in order to isolate problems with script execution. Execution of non-contiguous commands also allows flexible use of existing scripts by allowing an operator to easily execute only a portion of a script or to execute most of a script but excluding some commands.

Please replace the paragraph that begins on page 12, line 11, with the following amended paragraph:

Two function buttons of the exemplary GUI display 200 support the command step functionality of the command dispatcher within the exemplary embodiment of the present invention, the Do All function button 232 and the Step function button 220. The exemplary embodiment allows an operator to select one or more commands within the command file 102. The selected commands may be contiguous or non-contiguous as is described in associated with the Do All function button 232. The command step functionality of the exemplary embodiment allows an operator to execute individual commands within [[the]] a group of selected commands. The step functionality begins by an operator selecting [[a]] the group of ~~command within~~ commands among the commands that are displayed within the command window 204. Upon selection of selected commands, which are a [[set]] group of commands that were selected by the operator from within the command window 204, the first instruction of the selected commands is indicated as the next instruction to be executed. Pressing the Step function button 220 causes execution of the command that is indicated as the next command to execute. After execution of the indicated command, the next command within the selected commands is indicated as the next command to execute and is executed when the Step function button 220 is next selected by the operator. For example, the first operation of the Step function button 220 after a set of commands within the command window 204 is selected causes the first command within the selected commands to execute. After the execution of that first command, the second selected command is then indicated as the next instruction to execute. Execution of the indicated command and indication of the next command within the selected commands continues until the last command within the selected commands is executed. The Step function button 220 allows the results of each command to be observed and/or evaluated prior to execution of the next selected command.

Please replace the paragraph that begins on page 15, line 4, with the following amended paragraph:

If the operator did not press the Undo All function button 234, the processing then advances to determine, at step 316, ~~if the operated~~ whether the operator pressed the Undo Step function button 224. If the operator did press the Undo Step function button 224, processing continues, at step 318, by executing the undo command that is associated with the "next" command. The next command is the command that is marked for execution, as is described herein.

Please replace the paragraph that begins on page 15, line 11, with the following amended paragraph:

If the operator did not press the Undo Step function button 224, the processing then advances to determine, at step 320, whether the operator pressed the Step function button 220. If the operator did press the Step function button 220, the next command in the selected command list is executed, at step 322. If the operator did not press the Step function button 220, nor any other function button that was previously tested in the above processing, the processing of the exemplary embodiment advances to ~~the handle other function buttons~~ a handle-other-function-buttons processing 400 (see FIG. 4).

Please replace the paragraph that begins on page 15, line 19, with the following amended paragraph:

The ~~handle other function buttons~~ handle-other-function-buttons processing 400 of the exemplary embodiment is illustrated in FIG. 4. Upon entry into the ~~handle other function buttons~~ handle-other-function-buttons processing 400, the processing determines, at step 402, whether the operator has pressed the Skip function button ~~232~~ 236. If the operator has pressed the ~~Skipped~~ Skip function button ~~232~~ 236, the processing causes the next command within the selected commands to be skipped. This is in contrast to the operation of the Step function button 220, which causes the next command to execute. Subsequent to pressing the Skip function button ~~232~~ 236, the selected command following the next command is indicated to be the next command to execute. Processing then returns to the exemplary GUI processing flow 300 to await further operator selections, at step 306.

Please replace the paragraph that begins on page 16, line 1, with the following amended paragraph:

If the operator did not press the skip function button ~~232~~ 236, the processing next determines, at step 406, whether the operator has pressed the cancel function button 228. If the operator has pressed the Cancel function button 228, processing advances to halt the execution, at step 408, of the currently executing command and script. The operation of the Cancel function button 228 causes that command to halt if the Step function button 220 had been previously pressed by the operator. If the operator has pressed the Do All function button 232, pressing the Cancel function button 228 causes the execution of the executing subset of the command file 102, i.e., the selected commands, to halt. The operation of the Cancel function button causes the command or script to be placed in a halted state. The processing then returns to the exemplary GUI processing flow 300 to await further operator selections, at step 306.

Please replace the paragraph that begins on page 16, line 29, with the following amended paragraph:

If the operator has not pressed any of the above described function buttons, processing continues by determining, at step 416, if the operator has pressed the Exit function button 230. Pressing of the Exit function button causes the GUI interface 106 to halt operation. If none of the above operator selections is determined to have been made, the operation of the exemplary GUI interface processing flow proceeds by handling the error due to the unrecognized command. Preferably, an error message is displayed on the screen and the ~~skip command~~ Skip function button ~~232~~ 236 can be default selected to pass control of the processing to the exemplary GUI processing flow 300 to await further operator selections, at step 306.